KERNEL LOADABLE WRAPPERS FOR INFORMATION ASSURANCE

Secure Computing Corporation

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-224 has been reviewed and is approved for publication

APPROVED:　　　　*Glen E. Bahr*

　　　　　　　GLEN BAHR
　　　　　　　Project Engineer

FOR THE DIRECTOR:　　　　*W. Debany*

　　　　　　　WARREN H. DEBANY, Technical Advisor
　　　　　　　Information Grid Division
　　　　　　　Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 2002	3. REPORT TYPE AND DATES COVERED Final May 99 – Mar 02

4. TITLE AND SUBTITLE
KERNEL LOADABLE WRAPPERS FOR INFORMATION ASSURANCE

6. AUTHOR(S)
Richard O'Brien, Terry Mitchem, Raymond Lu, and Ryan Leonard

5. FUNDING NUMBERS
C   - F30602-99-C-0123
PE - 33140F
PR - 7820
TA - 09
WU - 05

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Secure Computing Corporation
2675 Long Lake Road
Roseville MN 55113

8. PERFORMING ORGANIZATION REPORT NUMBER

00-0933415A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/IFGB
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2002-224

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*
Employing kernel loadable wrappers is an approach for selectively securing operating system functionality to provide robustness and security. Kernel loadable wrappers are loadable kernel modules that hook into system interfaces to perform pre-call and post-call processing. Their key features are that they are unbypassable, since they are in the kernel, and they are easy to install, requiring no modification to the operating system kernel. They can be used to make selected components more robust or to perform various security functions, such as fine-grained mandatory access control and monitoring network connections.

14. SUBJECT TERMS
Kernel Loadable Wrapper, Windows 2000, File System Wrapper, Network Wrapper, Access Control, Microsoft Media Player, Netscape Web Browser, Microsoft IIS Web Server, IISHACK, Kernal Hypervisors, STREAMs

15. NUMBER OF PAGES
42

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

Figures and Tables

iii

1 Document Overview

This document is the final report for the Kernel Loadable Wrappers (KLW) program sponsored by Air Force Research Lab (AFRL). It describes the technologies developed on the program, summarizes the major achievements, and documents lessons learned and open issues.

1.1 Introduction

Employing kernel loadable wrappers is an approach for selectively securing operating system functionality to provide robustness and security. Kernel loadable wrappers are loadable kernel modules that hook into system interfaces to perform pre-call and post-call processing. Their key features are that they are unbypassable, since they are in the kernel, and they are easy to install, requiring no modification to the operating system kernel. They can be used to make selected components more robust or to perform various security functions, such as fine-grained mandatory access control and monitoring network connections.

Kernel loadable wrappers are an outgrowth of Kernel Hypervisors [1] which were developed by Secure Computing Corporation under a pervious contract. Kernel loadable wrappers were originally implemented on Windows NT 4.0, but were migrated to Windows 2000 when it became apparent that Windows 2000 was going to replace Windows NT 4.0 as Microsoft's high- end operating system.

Kernel loadable wrappers are distinguished by the fact that they consist of loadable kernel code that is used to wrap specific operating system components. Other approaches have been used to provide wrappers for additional security or robustness including:

- Traditional hypervisors that replace the standard hardware interface with an interface that provides additional capabilities. This approach was used by Bressoud and Schneider [2] for building a replication hypervisor that intercepts, buffers, and distributes signals from outside the system.

- Special libraries that include security functionality and that are linked with an application before it is run. This is the approach taken by SOCKS [3] where the client links in the SOCKS client library. SOCKS is intended for use with client/server TCPIIP applications, but the concept of linking in special libraries can be used for any type of application.

- Userspace library wrappers that control access to standard library routines. This approach was taken by Bob Balzar on DARPA's Information Assurance (IA) program [4]

- Systems that wrap system calls inside a Unix kernel. This approach was taken by Secure Computing on the Kernel Hypervisors program [1] and by NM Labs on their IA wrappers program [5]

- Wrappers that make use of an operating system's debug functionality. This is the approach taken by the Berkeley group [6]

Kernel loadable wrappers have a number of benefits.

- They are unbypassable. Since the wrapper is implemented within the operating system kernel, malicious code cannot avoid the wrapper by making direct calls to the operating system as could be done with code library wrappers. Any such calls will be intercepted and monitored.

- They do not require kernel modifications. All kernel loadable wrapper code is implemented as modules that are loadable within the kernel while the system is running. There are no changes to kernel source code as is necessary with specialized secure operating systems.

- They are flexible. Kernel wrappers can be used both to implement a variety of different types of security policies and they can be applied to a number of different operating system components.

1.2 Accomplishments

The main accomplishment of this program was to develop an effective kernel based wrapper system for Windows 2000. This involved designing and implementing the following specific components.

- A kernel-resident Master Controller for managing policy and making security decisions

- A file system filter driver for monitoring file accesses

- A network filter driver for monitoring network accesses

- A management GUI for defining policy.

In addition, an install package for easy installation was created, and a User Guide was developed that described how to manage and use the KLW software.

1.3 Document Organization

The remainder of this document is organized as follows.

- Section 2 presents an overview of the KLW architecture, and Section 3 then presents a more detailed description of the design.

- Section 4 presents the evaluation results and the lessons learned.

- Section 5 contains a list of cited documentation.

- The Appendix contains the User Guide.

2 KLW Architecture

This section presents a high level overview of the KLW Architecture and then discusses some possible uses of the technology.

2.1 Overview

The KLW architecture is illustrated in Figure 1 on the next page.

There are five main components:

2

- **The master wrapper:** Acts as a security server and provides an interface that other wrappers can use to perform security checks. These checks use data from the KLW security database to return a yes/no decision when queried about requested accesses. The master wrapper also communicates directly with the KLW process monitor to keep track of processes that are created and destroyed, including what the parent/child relationships are.

- **The file system wrapper:** Controls access to file system resources based on the executable that is trying to access the resource and the pathname of the resource.

- **The network wrapper**: Controls network access based on the executable that is trying to access the resource and certain attributes of the requested resource. Attributes include local IP address, remote IP address, remote and local port number and others.

- **The process monitor**: Informs the master wrapper when a process is created or destroyed. In the case of process creation, it supplies the process ids (PIDs) of the parent and child processes, as well as the full pathname of the executable file that the child will execute.

- **The management component:** Provides a way for the user to communicate with the master wrapper. The types of communication include dynamically loading new security policies and making changes to the KLW configuration.

The details of the KLW design can be found in Section 3.

User Space

KLW Management

File Access

Application

Network Access

KLW
File System
Wrapper

KLW
Process
Monitor

KLW
Network
Wrapper

Access
Check

Access
Check

KLW
Master
Wrapper

Win2k
File
Systems

Win2k
Network
Stack

KLW
Security
Database

Kernel Space

- KLW master wrapper provides access check functions based on policy stored in the Security Database
- KLW file system wrapper controls access to standard Win2k file systems
- KLW network wrapper controls access to the standard Win2k network stack
- KLW process monitor monitors process creation/destruction and provides information to the master wrapper
- KLW management provides a way for the user to dynamically load the KLW security policy and change the KLW configuration

Figure 1 KLW Architecture

2.2 Types of Uses

Kernel loadable wrappers can be used in a variety of ways to enhance the security and robustness of a system.

2.2.1 Fine-grained mandatory access control:

The most compelling use of kernel loadable wrappers is to provide dynamic, mandatory fine grained access control to various system resources such as files and network sockets. The type of control possible is what differentiates this method of wrapping from standard access control list methods that perform control based on a user attribute. Kernel loadable

4

wrapper security policies can be based on users, but can also be based directly on the application.

2.2.2 Auditing:

In their simplest form, kernel loadable wrappers can provide an audit and monitoring functionality that merely records additional information about the system resources that a process is accessing. Such audit wrappers are also useful for determining what system resources an application accesses during its normal processing.

2.2.3 Label-based access control:

While the rule sets described later can be used to implement most simple policies, it would also be possible to use kernel loadable wrappers to implement more sophisticated label-based policies, such as a multilevel secure (MLS) policy or a type enforcement policy. To be able to implement these policies, a kernel wrapper would need a way to label system resources. This might be done by creating and maintaining its own list of labeled names, or by piggybacking the labels on system data structures that have available space. Because of the flexibility of the kernel loadable wrapper, these label-based policies could be applied to only those portions of the system that required labels. And policies could be quickly changed, if needed, to adapt to the current operating environment.

We have investigated several ways to implement file labeling on Windows NT and Windows 2000 and have found that the STREAMs feature of the NTFS 2000 file system (which was added in Win2K) provides the necessary functionality. The STREAMs feature of NTFS 2000 allows more than one stream of data to be associated with any particular file. Using this feature, it would be possible to add a named stream to a file for the purpose of holding security labels. We have also looked at using NTFS extended attributes to hold security labels since this would allow operation on Windows NT as well as Windows 2000, but the amount of complexity and performance overhead associated with this method would be prohibitive.

3 High Level Design

This section documents the high level design of the kernel loadable wrappers system.

Kernel loadable wrappers were implemented on the Windows 2000 kernel using the concept of a kernel loadable module. Win2K was chosen as the initial platform because it supports loadable modules, and because it is one of the world's most popular and widely used server operating systems.

As shown in Figure 1 in Section 2, there are five major components of the KLW system:

- The Master Wrapper
- The File System Wrapper
- The Network Wrapper
- The Process Monitor

- KLW Management Tool.

The following sections describe these components in more detail.

3.1 KLW Master Wrapper

The KLW Master Wrapper (KLWMW) or controller is a kernel-mode device driver without a physical device. The KLW controller can be loaded and unloaded by a user program on the

Windows-2000 operating system. The primary functions of the KLW controller are:

- Communicating with clients

- Processing ACL configurations

- Receiving process information

- Validating access.

3.1.1 Communicating with Clients

The KLW controller is the central management component that must communicate with all of the other components. As shown in Figure 1, the File System Wrapper and the Network Wrapper call the KLW controller to request access decisions; the Process Monitor calls the KLW controller to report on process creations and deletions (which the controller records); and the Management Tool calls the KLW controller to update the access control and audit policies that are being enforced.

The communication between the KLW controller and the other client components is done via the use of 1/0 Request Packets (IRPs). An IRP is a data structure generated by the 1/0 manager for controlling how the 1/0 operation is processed. The function code field in a packet tells the KLW controller about the actual request of the IRP. The supported function codes are:

SET_POLICY: used by the Management Tool to install/modify the policy databases

- NOTIFY_IMAGE_LOAD: used by the Process Monitor to pass the KLW controller the mapping between process ID and program executable when an executable is loaded

- NOTIFY_PROCESS_START: used by the Process Monitor to pass the KLW controller information about a new process that has started

- NOTIFY_PROCESS_CLOSE: used by the Process Monitor to inform the KLW controller that a process has terminated

- VALIDATE_FILE_ACCESS: used by the File System Wrapper to request an access decision on a file system access

- VALIDATE_NETWORK_ACCESS: used by the Network Wrapper to request an access decision on a network access.

3.1.2 Processing ACL Configurations

The access decisions that the KLW controller makes are based on the access policy rule database that it maintains. When the KLW Controller gets loaded into the system, it initially does nothing, waiting for the KLW management tool to send in the initial access control configuration information. The access control information includes the application programs that need to be monitored and their access policy. Whenever the KLW controller receives a new and valid access configuration from the management tool, it updates its policy rule database with the new information and discards the previous rules.

3.1.3 Receiving Process Information

Once the policy rule database is populated, the KLW Controller starts monitoring the processes generated directly and indirectly by the configured application programs.

The KLW controller receives process-related information from the Process Monitor via an IOCTL function. This allows the controller to be informed when processes are created or destroyed and allows it to maintain a record of parent/child process relationships. The controller is also informed as to what executable file is being used by a particular process.

3.1.4 Validating Access

The KLW controller is ready to validate an access request after it creates the rule database and begins monitoring process creations. The KLW clients can then send in requests to validate file or network access for a particular process. Upon receiving a request sent by a client to validate the access privilege of a process, the controller first verifies if the particular process is being monitored. It then checks the access policy of the process and decides if the access should be granted or denied. The access policy of a parent process also applies to its child processes.

3.2 KLW File System Wrapper

The function of the KLW File System Wrapper component is to enforce mandatory file access rules for all access to the NTFS File System. It does this by communicating with the master wrapper to perform access checks and using the results of the checks to determine if a file system operation should be allowed.

3.2.1 Overview

The function of the KLW File System Wrapper (KLWFSW) is to intercept certain 110 functions that are handled by the Win2K NTFS driver and request that the KLW controller validate the requested operation. After receiving a decision from the controller, the KLWFSW allows or denies the 110 request based on the answer received from the controller. Figure 2 illustrates the basic architecture of the KLWFSW.
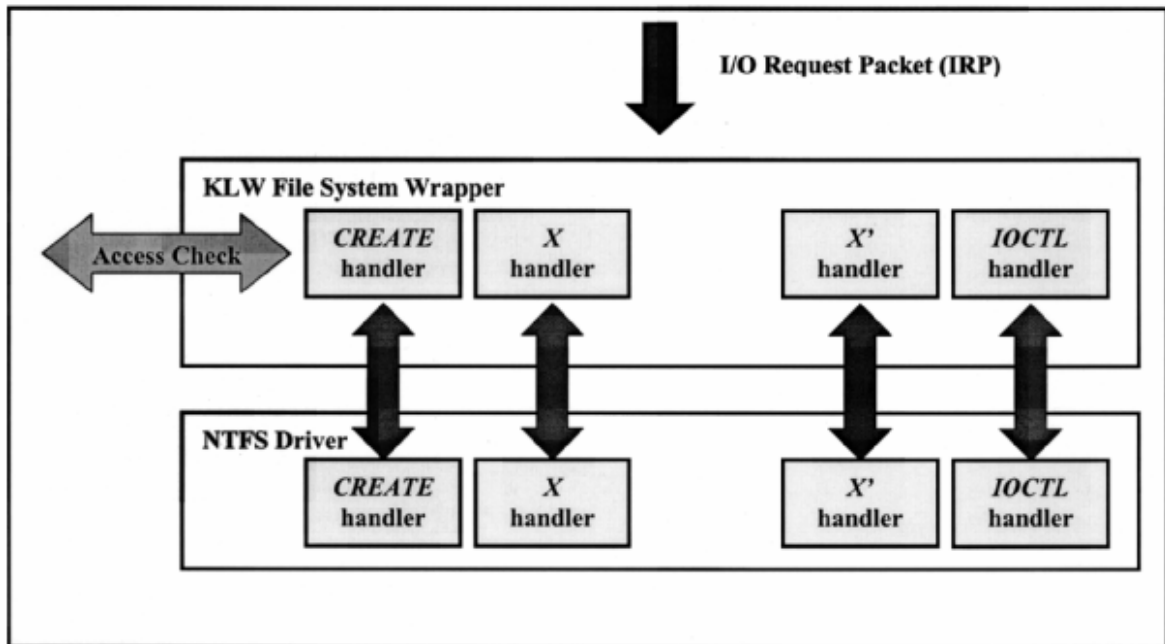
Figure 2 File System Wrapper Architecture

3.2.2 Filter File System Driver Implementation

The KLWFSW is implemented as a Windows 2000 filter file system driver. Since the source code for NTFS is not available and reverse engineering would be extremely difficult, it is necessary that the KLWFSW design use documented kernel interfaces that are supplied (and approved) by Microsoft. Filter file systems are standard Win2K system components that can provide value-added functionality to the kernel. Some examples of such value added functionality are on-the-fly encryption and virus detection. Filter file systems are also used extensively by Microsoft to implement remote file sharing.

3.2.3 Processing Flow

The steps that are performed by the KLWFSW during normal operation are as follows:

1. Receive an IRP in the dispatch routine

2. Determine if the IRP is destined for the NTFS driver or the KLWFSW itself.

3. If the IRP is for the KLWFSW

    a. Process the request and return, else

4. If the IRP type is other than IIRP_MJ_CREATE, pass the IRP through, else

5. Gather additional information (such as the fully qualified pathname) about the filename if necessary

6. Send a security check request to the KLWMW to find out if the operation should be permitted

7. If the operation is permitted, pass the IRP on to the next driver in the stack, else complete the IRP immediately with an error.

3.2.4 File System Controls

Access control policies are defined by lists of access control rules (called ACLs) that are passed into the Master Controller and used by it to make access control decisions.

For file system operations, a specific access control rule contains the following fields:

- Monitored program name
- User name

    Target path/file name

- Access privilege
- Audit choice.

The *Monitored program* name identifies the program executable that is being wrapped. All accesses by this program to the *Target path/file name* will be monitored and only those accesses that are specifically given in the *Access privilege* field are allowed. The possible accesses are: file_read(**R**), file_write(**W**), file_execute(**E**), or some combination of these three, and no_access(**N**). The *Audit choice* indicates whether a match on this rule should be audited. There is additional control over what is actually audited at the global level (audit on/off) and at the program level (audit all, only allows, only denies, no audit). These additional controls are discussed more completely in the KLW User Guide (see Appendix A). In the current implementation, the *User name* field is not used. Its purpose is to allow finer control based on which user is running the program, but this additional control has not yet been implemented.

The format of an ACL rule and an example of an actual line from the file section are shown in Figure 3. The example states that the executable **\VSCS\bin\vscs.exe** running as any user has read access to the file C:\VSCS\conl\config-sav.txt and all read accesses by the program to the target file should be audited.


```
        <program name><user id> <target> <access> <audit>

\VSCS\bin\vscs.exe    *        C:\VSCS\conf\config-sav.txt R      y
```
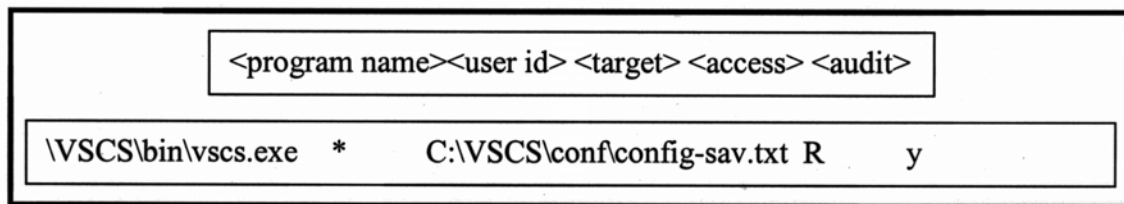
Figure 3 Sample File System ACL

It is important to note that a monitored program name can be specified with a parent program name so that a policy can be applied to a process of a specific program spawned from a process of another specific program. An example of one of these rules is shown in Figure 4. This rule states that the executable **\VSCS\bin\admin.exe,** when spawned by

9

the executable **\VSCS\bin\vscs.exe,** has read/write access to the file
C:\VSCS\conf\config-sav.txt and accesses should be audited.

```
\VSCS\bin\vscs.exe->\VSCS\bin\admin.exe * C:\VSCS\conf\config-sav.txt RW   y
```

Figure 4 Sample Child Process ACL

3.2.5 NTFS file access modes vs. KLW file permissions

Currently, security checks are performed only on IRP_MJCREATE IRPs. This is because Win2K requires that a program ask at open time for the maximum privilege that it requires for the time that the file will be open. When the KLWFSW receives the "create IRP", it converts the NTFS access mode to a KLW file permission and then performs the check. Currently, the KLW file permissions are limited to read, write and execute, while NTFS supports numerous access modes. This leads to a significant amount of overloading when mapping NTFS access modes to KLW permissions. The mapping of NTFS modes to KLW file permissions is shown in Table 1. The number of KLW file permissions was kept low in order to simplify the design and security database. In the future, it may be desirable to expand the number of KLW file permissions so that the KLW policy can provide granularity closer to that of NTFS.

NT Access	KLW Permission
WRITE_DAC	KLW_FILE_WRITE
WRITE_OWNER	KLW_FILE_WRITE
DELETE	KLW_FILE_WRITE
VILE_READ_DATA	KLW_FILE_READ
VILE_WRITE_DATA	KLW_FILE_WRITE
VILE_APPEND_DATA	KLW_FILE_WRITE
FILE_READ_EA	KLW_FILE_READ
FILE_WRITE_EA	KLW_FILE_WRITE
FILE_EXECUTE	KLW_FILE_EXECUTE
FILE_READ_ATTRIBUTES	KLW_FILE_READ
FILE_WRITE_ATTRIBUTES	KLW_FILE_WRITE

Table 1 Filesystem Access Mapping

3.3 KLW Network Wrapper

The function of the KLW Network Wrapper (KLWNW) component is to enforce mandatory network access rules for all access to the TCP, UDP and Raw IP network protocols. It does this by communicating with the master wrapper to perform access checks and using the results of the checks to determine if a network operation should be allowed.

3.3.1 Overview

The KLW Network Wrapper intercepts certain network-related I/O that is handled by the Win2K TCP, UDP and Raw IP drivers and requests that the KLW controller validate the requested operation. After receiving a decision from the controller, the KLWNW allows or denies the 110 request based on the answer received from the controller. Figure 5 illustrates the basic architecture of the KLWNW.

3.3.2 TDI Filter Driver Implementation

The KLWNW is implemented as a standard Windows 2000 filter driver. Certain I/O requests bound for the three filtered drivers are intercepted and extra processing is done to record information and perform security checks. Since the drivers that the KLWNW filters are Transport Device Interface (TDI) drivers, the KLWNW supports this interface, which is primarily accessed through IOCTL I/O requests. TDI is the standard interface that all protocol drivers in Win2K use.
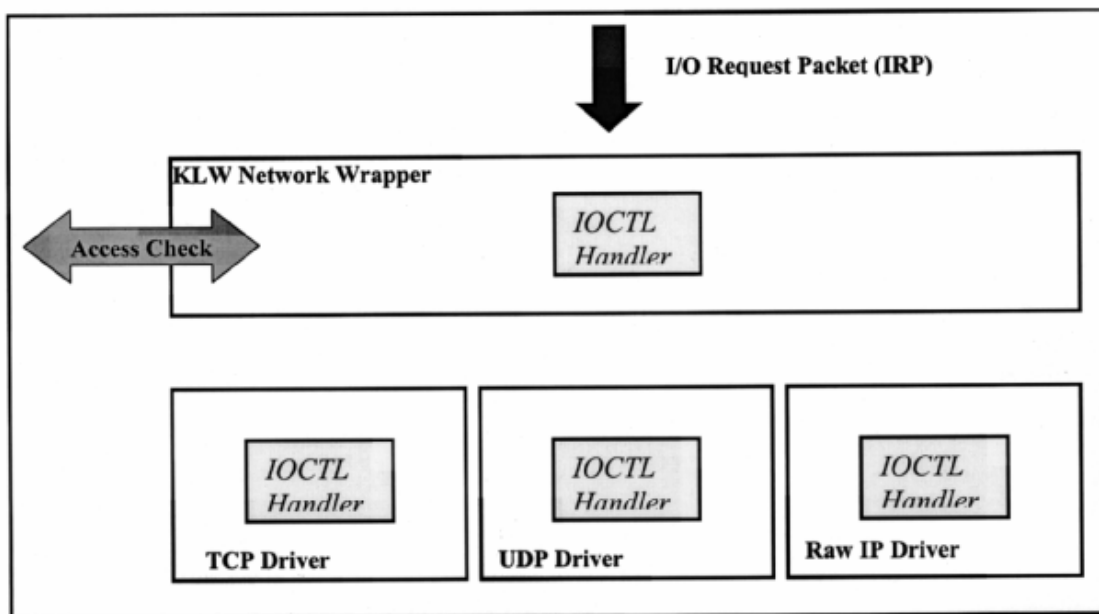
Figure 5 Network Wrapper Architecture

3.3.3 Processing Flow

Creating a socket using TDI is a very complicated process. A single socket() call done in user space with the Winsock API will result in numerous TDI calls. The list below shows

11

what happens (in a general TDI sense) when a user calls a typical socket()/connect() sequence to initiate an outgoing connection.

- socket()

  o TdiAddressOpen() — Called to create a TDI address object for the local address connect()

  o TdiAddressOpen() - Called to create a TDI address object for the remote address.

  o TdiCreateConnection() — Creates a TDI connection object

  o TdiAssociateAddress() — Binds the local address to the connection object.

  o TdiAssociateAddress() — Binds the remote address to the connection object.

  o TdiConnect() — Initiate the connection described by the connection object.

Only certain calls in the list above are of concern to the KLWNW. The TdiCreateConnection() call is filtered so that the location of the connection object can be stored for later use when the TdiConnect() call is made. When TdiConnect() is called, the KLWNW extracts the local and remote address information from the connection object and performs a security check to the KLWMW to see if the connection should be allowed.

The processing that occurs when accepting an incoming connection is similar but is made much more complex by the use of TDI event handlers that get invoked when an incoming connection actually arrives. A description of this process is beyond the scope of this document. Those wishing for a complete description should refer to the Windows 2000 DDK documentation and the TDI Sample Drivers distributed by PCAUSA.

3.3.4 Network Controls

For network operations, a specific access control rule contains the following fields:

- Monitored program name

- User name

- Local]P Address

- Remote 1P Address

- Port/Port Range

- Direction

- Audit choice.

The *Monitored program name* identifies the program executable that is being wrapped. All incoming and outgoing TCP/UDP network traffic by this program is monitored and any packets that match the *Local IP Address, Remote IP Address*, and *Port/Port Range* are filtered based on the *Direction* field. The possible values of the Direction field are: i**n, out, both** or **none.** The *Audit choice* indicates whether a match on this rule should be

audited. As in the case of file system operations, in the current implementation, the *User name* field is not used.

The filtering works as follows. For incoming traffic, if the source IP address in the packet is the *Remote IP Address* and the destination IP address is the *Local IP Address* and the destination port matches the *Port/Port Range* value, then the rule has been matched and the *Direction* is checked. If the Direction value is **in** or **both**, then the traffic is allowed. Otherwise, the traffic is not allowed. Similarly for outgoing traffic, if the source IP address in the packet is the *Local IP Address* and the destination IP address is the Remote IP Address and the destination port matches the *Port/Port Range* value, then the rule has been matched. If the Direction value is **out** or **both**, then the traffic is allowed. Otherwise, it is not.

An example of a network rule is shown in Figure 6.
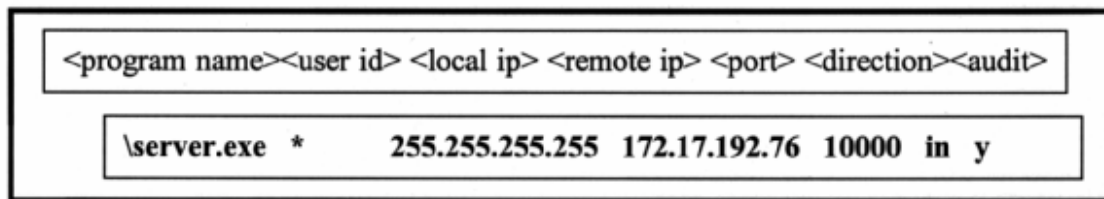
Figure 6 Sample Network ACL

This rule states that the program whose executable is **\server.exe** is allowed to receive broadcast packets (destination = **255.255.255.255**) on port **10000** that originate from the IP address **172.17.192.76.**

3.4 KLW Process Monitor

The KLW process monitor tracks the creation and destruction of processes. It does this by using Windows 2000-specific interfaces that have been provided by Microsoft for driver writers who wish to know when processes are created and destroyed. Unfortunately, these routines do not provide any way for a driver to deregister itself after using them. This means that using these routines and then unloading the driver that used them will crash the system. The process monitor allows us to load and unload dynamically the majority of the KLW code.

Instead of hooking relevant system calls, the KLW controller takes advantage of two Windows- 2000 specific system notification routines to help monitor processes. The routines are *PsSetCreateProcessNotifyRoutine()* and *PsSetLoadImageNotifyRoutine()* The KLW Controller calls *PsSetCreateProcessNotifyRoutine()* to register itself with the operation system so that it would receive a system notification whenever a process is created or deleted. It also calls PsSetLoadlmageNotifyRoutine() to receive a system notification whenever a file image is loaded for execution. With the help of these two system notifications, the KLW controller is able to determine which processes need to be monitored based on the configuration information. Once the registration with the

Windows 2000 system is done, the notification remains effective until the system itself is shut down. In other words, a system reboot is required to test the modified program, which contains the source code for system notification registration. For this reason, the notification portion of the KLW code needs to be implemented as a separate driver so that any modification of the rest of the KLW controller code does not require restarting the system.

3.5 KLW Management Tool

The KLW management tool is an application program in user space. It allows the users to configure dynamically the access policies over system resources, including file system and networks, with the KLW controller. It also allows the users to start (load) and stop (unload) the KLW device drivers, including the KLW controller and the KLW clients.

3.5.1 Management Interface

The KLW management tool is a Java GUI application that runs in user space. A complete description of the user interface is given in Appendix A.

3.5.2 Communication with KLW Controller

To communicate with the KLW controller, the management application calls OpenDevice() with a designated KLW device name, klwmasterDeviceO to get a handle back from the system. It then calls DeviceloControl() with the handle, the function code, and other information to send requests to the controller.

4 Evaluation

To evaluate the effectiveness of the KLW technology, an initial analysis was performed that identified strengths and weaknesses of the current implementation. The results of this analysis are summarized in Section 4.1. Some limited penetration testing was also done. Three applications were wrapped and various attacks run against them. The three applications were the Microsoft Media Player, the Netscape Web Browser, and the Microsoft IIS Web Server. The results of this testing is described in Section 4.2. In general, the testing validated the effectiveness of KLWs. Section 4.3 contains a summary of the lessons learned.

4.1 Analysis Results

The current KLW system can monitor and control two types of operations.

File System operations using the standard interfaces can be controlled based on the program performing the operation and the target file.

Network operations using the TCP and UDP protocols can be controlled based on the program name and the target IP address and port.

In general, there are two common approaches to using these controls on a host system: application-centric and resource-centric. Section 4.1.1 discusses the application-centric approach to using KLWs. Section 4.1.2 discusses the resource-centric approach. Section 4.1.3 identifies some limitations of the current implementation.

4.1.1 Controlling applications

The approach is most suitable when the user wants to control a program that may have, or is likely to have, vulnerabilities. Some programs that fall into this category are email programs, web browsers, and network-enabled media players such as Microsoft Media Player and RealPlayer. The number of different attacks against such programs is large. The attacks range from malicious macros buried in email message to buffer overrun attacks that are designed to cause the targeted program to perform actions that is was not designed to do.

While KLWs cannot necessarily protect the vulnerable program from being compromised, they can be used to protect the rest of the system from the actions of a compromised program. This is done by using the KLW file system and network controls to strictly limit the operations that the program (and any of its children) can do. In particular, by identifying what files the program needs to access during its normal operation and only allowing it access (with the appropriate R, W, X permissions) to those files and no others, the remaining files on the system are protected from being read or modified by the program, even if it has been compromised. Similarly, by identifying what network connections the program needs during its normal operation and only allowing those connections for the program and its children, illegal network activity can be prevented.

4.1.2 Controlling access to resources

This approach is most suitable when the user has specific data to protect and does not know from where an attack might originate. A web server-based application that stores sensitive data (such as credit card numbers) on the host machines file system is an ideal candidate for this approach. The files or directories that contain the sensitive data can be declared off-limits to all programs except those applications that specifically need to access the data. This prevents other programs from accessing the data even if they are compromised. If the number of applications that need access to the data is low, this can result in a smaller, simpler security policy. This approach will not prevent vulnerable programs on the host from being corrupted, but it will strictly limit access to the sensitive data that the user wished to protect.

If the user uses this approach, combined with application-specific controls for the programs that must access the data (if they might be vulnerable), this will result in very effective security.

4.1.3 Limitations of the Current Implementation

Since the current KLW implementation only monitors and controls standard file system and network access, it is incapable of stopping certain other classes of attacks. This section will attempt to describe some of these classes of attacks and suggest enhancements to the current KLW implementation so that it will be capable of stopping them.

Registry-based attacks. Registry based attacks are a common type of attack used on Windows platforms. The registry is generally used to hold configuration information for

the operating system and third party applications, but almost any sort or information for any program can be stored there. In the case of some poorly designed programs, security-critical information is kept there. This makes a registry-based attack a good choice for use against these programs. It is fairly easy for malicious code to change a program's configuration information, or even to change the operating system configuration.

It would be fairly easy to implement registry control in KLWs. Freely available software, such as NT Regmon from Sysinternals.com, comes with the necessary kernel code to implement this control. NT Regmon uses a system call hooking technique that allows it to monitor all system calls that perform registry operations.

Local IPC attacks. The Windows platform provides several ways for applications running on the same machine to communicate with each other. It is known that some programs that use these local JPC mechanisms are vulnerable to buffer overrun attacks through those mechanisms. It might be possible to hook the systems calls that provide the interface for local IPC in order to enforce a security policy on them, allowing KLW to decide which applications should be allowed to communicate with each other. This would be a valuable feature to add to KLW and research needs to be done in this area.

File system mount point attacks. One of the weaknesses of the KLW file system security model is its dependence on pathnames in the security policy. Since program names and target filenames are specified with absolute pathnames, any action that moves a file system from one mount point to another will void the KLW security policy if the policy is being applied to files on that particular file system.

A registry based attack that causes a file system to be mounted at a different mount point will most likely succeed because it will void the KLW security policy. The enhancements described in section 4.1 would allow KLW to stop this sort of attack. By preventing mount points from being changed, the KLW security policy would always remain correct.

Service mechanism attacks. The service mechanism on Windows 2000 is designed to allow a process to run as a special kind of process called a service. Services are analogous to daemons on a UNIX system, and generally perform system level tasks such as authentication and general system upkeep. On Windows 2000, services also handle the loading and unloading of dynamic kernel code. This means that any process running with Administrator privileges is capable of inserting code into the kernel. This capability would allow an attacker to circumvent all the KLW security measures as well as the standard Windows 2000 security policy. Once an attacker has access to the kernel, they can do **anything** that they want. It is possible that hooking certain system calls would let KLW control this process.

4.2 Wrapped Examples

Three particular examples were wrapped using the KLW technology developed on the program: Microsoft's Media Player, Netscape's Web Browser, and Microsoft's Internet Information Server (IIS). The goal of the wrapping was to prevent attacks that might be propagated via the network and, in particular, web based applications.

4.2.1 Microsoft Media Player

To evaluate the effectiveness of KLW at stopping attacks directed at Media Player, Secure tried two attacks that were available on the web. The attacks were first executed against an unwrapped version of Media Player and then against a version wrapped by KLW.

The first attack against the unwrapped version was a buffer overrun attack that attempted to write a file in the root directory of the C: drive. The results showed that this attack was successful. Secure then executed the same attack against the wrapped version of Media Player, which only allowed the Media Player access to those files that it needed to operate, and the attack failed. Since the security policy enforced by KLW specifies which files that Media Player has access to and what those files are, the attack was stopped since the security policy disallowed write access to c:\. Since all file system access through the standard interfaces can be controlled by KLW, it is doubtful that any file system-based attack will succeed if the KLW security policy is correct.

The second attack executed against the unwrapped version caused Media Player to perform the following sequence of actions:

1. Begin displaying a movie.

2. Approximately 1 minute into the movie, start a copy of the users default browser and instruct it to go to a particular website.

3. Approximately 1 minute after that, the browser is directed to another website.

4. Approximately 30 seconds after that, the browser is directed to a webpage that is designed to crash the system the browser is running on. This page attempts to use the /dev/nul device as an html image.

There are two ways that KLW can be used to stop this attack:

1. The policy can be configured to prevent Media Player from launching this browser. This solution works, but it is not optimum because launching the default browser is a common operation for Media Player.

2. The policy can be configured to prevent the launched browser from accessing /dev/nul. This solution is preferable because is does not limit the functionality of Media Player.

KLW's ability to control program execution (via the file system) should be effective in stopping attacks that rely on the attacked program launching another vulnerable program.

4.2.2 Netscape Navigator

To evaluate the effectiveness of KLW at stopping attacks directed at Netscape Navigator, Secure exercised certain capabilities of Navigator that are normally permissible but could be abused by an attacker. This functionality was first exercised with an unwrapped version of Navigator and then with a version wrapped by KLW.

The experiment that Secure conducted was to have Navigator (running as the *Administrator user*) attempt to save files and overwrite critical system files. Saving files from Navigator is functionality that every average user needs, but the location that the

files are saved in and the names they are given is very important. It is quite possible for an inexperienced user or a corrupted version of Navigator to destroy critical system files. When Secure tried this, we were able to overwrite any file that we chose. After wrapping the program with KLW, we were able to deny access to critical files. It is important to note that there are files that are part of the Netscape Navigator distribution which the Navigator executable must be able to write, such as the bookmarks file, which could be maliciously modified by an attack. These attacks might take the form of invisibly modifying the UIRL associated with a particular bookmark to direct the user to some undesirable location. Unfortunately, these sorts of attacks cannot be stopped without limiting Navigator's functionality to an unacceptable degree.

4.2.3 Microsoft Internet Information Server

Recently a serious new bug was discovered in Microsoft's Internet Information Services (IIS) version 5.0 for Windows 2000. To test the effectiveness of KLWs, we decided to wrap 11S and see if the wrapper could protect against attacks based on exploiting this bug. The bug consists of an unchecked buffer in the section of the Internet Printing Protocol (IPP) Internet Server Application Programming Interface (ISAPI) that handles input parameters. By overrunning this buffer with malicious code, an attacker can execute code of their choice on the server in the Local System security context. More details about the flaw can be found at:

http://www.microsoft.comltechnetlsecurity/bulletin/MSO 1-023 .

IISHACK2000.c, by Ryan Permeh, is a hack that takes advantage of this flaw. It can be found at http://www.hack.co.za. More information about the hack itself can be found in the comments of IISHACK2000.c. IISHACK overruns the buffer in the IPP ISAPI with malicious code that then attempts to write to the C:\ drive in the Local System security context. When run against an unprotected Windows 2000 machine using US 5.0, a text file named www.eEye.com.txt will be created in the C:\ drive.

Using KLW, we created a "basic" policy for US that allows it to do its normal tasks but nothing else. This policy was specifically denied write access to anything but certain log files. When IISHACK was run on the protected computer the attack was stopped. While the buffer was overrun (the overrun itself exploits normal 11S behavior and so cannot be stopped without losing US functionality) the text file was not written to the C:\ drive. So while the malicious user would have been able to get code running in the Local System security context, they would not have been able to do anything 11S is not expected to do to the file system.

4.2.4 Summary

The results of the penetration testing validated the analysis described in Section 4.1. That is, the current KLW implementation is effective against attacks that require access to files that can be protected from the application. We did not attempt any of the type of attacks described in Section 4.1.3. The analysis indicates that these attacks might succeed against the current implementation, as long as they do not require a file system access that we can control.

4.3 Lessons Learned

Since the current KLW implementation only monitors and controls standard file system and network access, it is incapable of stopping certain other classes of attacks. In particular, on this program no work was directed at protecting the registry or monitoring system calls. As a result, attacks that might modify the registry and/or attacks that are based on using certain system calls are not stopped by the current implementation of KLWs, as described in the previous section. This is not a shortcoming of the technology; rather it is a consequence of time and schedule limitations on the program. Techniques are known that would allow system calls to be intercepted and controlled. Any transition of the technology should include the work necessary to add this additional functionality.

The current file system wrapper is based on file system pathnames. As a result, changing the path name of a file may change the effectiveness of the policy. In the short run, this can be addressed by ensuring that critical files (and their attributes) cannot be modified except by appropriate applications. In the long term, a better solution would be to assign security labels to the files (as discussed Section 2.2.3) rather than pathnames and to use these immutable types in any access decisions. This would ensure that even if the filename were changed, accesses to the file would still be properly enforced.

4.4 Conclusion

The conclusion of our evaluation is that KLWs can provide an effective deterrent against a wide variety of known and unknown attacks against Windows 2000 applications. While the current version of KLWs only monitors and controls file and network accesses, extending the KLWs to monitor and control system calls, in particular those that access the registry, those that provide inter-process communication, and those that mount file systems, would add significant capabilities.

5 References

[1] Terrence Mitchem, Raymond Lu, Richard O'Brien. Using Kernel Hypervisors to Secure Applications. Proceedings of the 1997 Annual Computer Security Applications Conference (ACSAC97), December 1997.

[2] Thomas Bressoud and Fred Schneider. Hypervisor-based Fault-Tolerance. Proceedings of the 15th ACM Symposium on Operating System Principles. December 1995. ACM Press.

[3] M. Leech, Ct al. RFC 1928: SOCKS Protocol Version 5. March 1996.

[4] Robert Baizer and Neil Goldman. Mediating Connectors: A Non-Bypassable Process Wrapping Technology. Proceedings of the DARPA Information Survivability Conference and Exposition, 2000 (DISCEXOO), January 2000.

[5] Timothy Fraser, Lee Badger, Mark Feldman. Hardening COTS Software with Generic Software Wrappers. Proceedings of the DARPA Information Survivability Conference and Exposition, 2000 (DISCEXOO), January 2000.

[6] Ian Goldberg, David Wagner, Randi Thomas and Eric Brewer. A Secure Environment for Untrusted Helper Applications. Proceedings of the 6th USENIX Security Symposium. July 1996.

Appendix A
KLW User Manual

A.1 Overview

The Kernel Loadable Wrappers program secures a system by setting and enforcing file and network access permissions. For any given executable you can control whether it can read, write, or execute to any given file. You could, for example, disallow Media Player write access to any files other than C:\Media\Save. You can also control what IP addresses and ports a given executable can send to and receive from. In this fashion you could disable Netscape from receiving packets from certain unknown or unsafe IP addresses or only allow it to listen on port 3042. These policies are written using a Graphical User Interface called the Kernel Loadable Wrappers Manager. This document describes the KLW Manager and how to create and maintain a KLW policy. Section A.2 looks at installation and startup; Section A.3 goes over each part of the manager in detail; and Section A.4 consists of a tutorial on creating a file and a network policy.

A.2 KLW Installation and Startup

A.2.1 Platform requirements

- Windows 2000 Server or Professional by Microsoft (

- Java 2 (Java Development Kit 1.2.1 or Java Runtime Environment 1.2.1) or higher by Sun Microsystems (installed

A.2.2 Installing Kernel Loadable Wrappers

You will need the Microsoft Installer (MS1) http://www.microsoft.com Windows 2000 comes packaged with MSL. Place the KLW CD into your drive and double click KLW.msi. The Microsoft Installer will lead you through the rest of the installation procedure.

A.2.3 Starting Kernel Loadable Wrappers

After installing, there will be a KLW entry in the Programs section of the start menu. Select this, and first click on KLW Readme to check that the configuration is correct for your system. You may need to edit the KLW batch file, startKlwGui, before continuing. Then click on KLW GUI to start the KLW Manager program. Under the first tab (Admin) you will find a button labeled "KLW is Off". You need to press the "KLW is Off" button in order to load the KLW drivers to your system and make the KLW Filter ready for work. The button will change to "KLW is On" and be grayed out. Once the KLW drivers are loaded, however, they cannot be unloaded. You need to restart your computer in order to undo the load command.

You can create a new policy folder or use an existing policy folder to build the file and network access rules. Once you are satisfied with the rules, you can create the policy database using the "Create Database" menu item under the file menu or the "Create

Database" button under the Alias tab. Then click on the "Load Policy" button in the Admin tab to enforce the access policy you have created in the current policy folder.

A.2.4 Viewing

The KLW audit can be toggled on and off by pressing the "Audit is On" button. The KLW audit messages are saved in the files under the \klw\audit\ directory. Audit cycles through a set of 20 files, each with up to 1000 records. You can view the messages with a text editor, such as notepad.

A.2.5 Reporting bugs

To report software problems, please send e-mail to: **mitchem@sctc.com** with a subject heading of "**KLW bug**".

A.3 The Kernel Loadable Wrapper Manager

This section documents the KLW graphical user interface, called the KLW Manager. Section A.3.1 gives a general overview of the Manager's purpose and sections A.3.2, A.3.3, A.3.4, A.3.5, A.3.6, and A.3.7 look at each section of the Manager individually.

A.3.1 Manager Overview The KLW Manager

- The File Menu
- The Help Menu
- TheAdminTab
- The Alias Tab
- The File Policy Tab
- The Network Policy Tab

The following sections describe these components in more detail.

A.3.2 File Menu
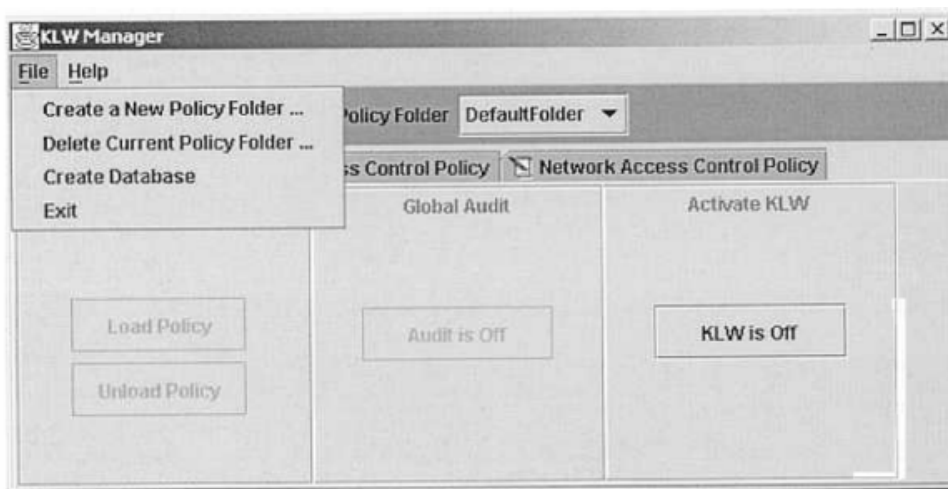
A.3.2.1 Add a New Policy Folder...

Selecting this will open the Add a New Policy Folder Dialog. A Policy Folder is used for storing the various files required to edit and enforce the policy you create. In order to switch between policies you need to switch policy folders.

A.3.2.1.1 The Add a New Policy Folder Dialog
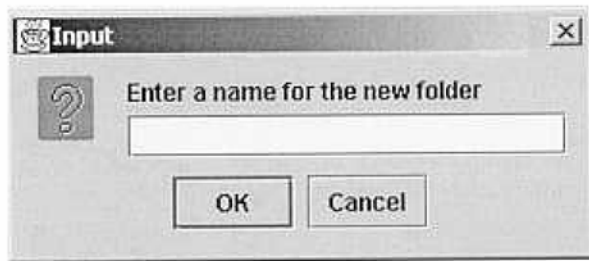
A.3.2.1.1.1 The "Enter a name for the new folder" Text Field.

Enter the name for the new policy folder here.

A.3.2.1.1.2 The "OK" Button

Click here to create the new policy folder.

A.3.2.1 .1.3 The "Cancel" Button

Click here to exit without saving.
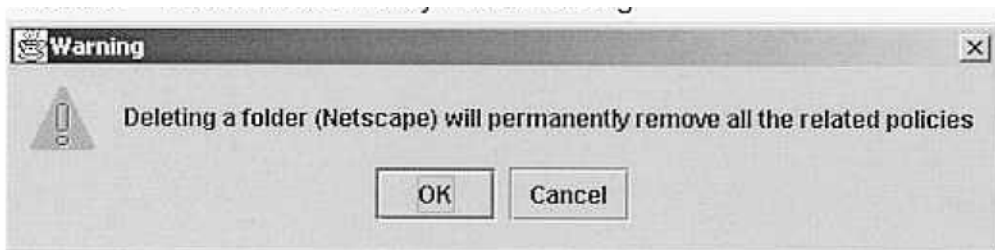
A.3.2.2 Delete a Policy Folder...

Selecting this will open the Delete a Policy Folder Dialog.

A.3.2.2.1 The Delete a Policy Folder Dialog

A.3.2.2.1.1 The "OK" Button

Click here to delete all the policy associated with the selected folder.

A.3.2.2.1.2 The "Cancel" Button

Click here to exit without deleting.

A.3.2.3 Create Database
Select this to create the policy database that can be loaded to the KLW driver.

A.3.2.4 Exit
This only exits the KLW Manager console. The KLW filter will not stop running.

A.3.3 Help Menu
Under the Help menu the About dialog tells you what version of the KLW Manager you are using.

A.3.4 Admin Tab
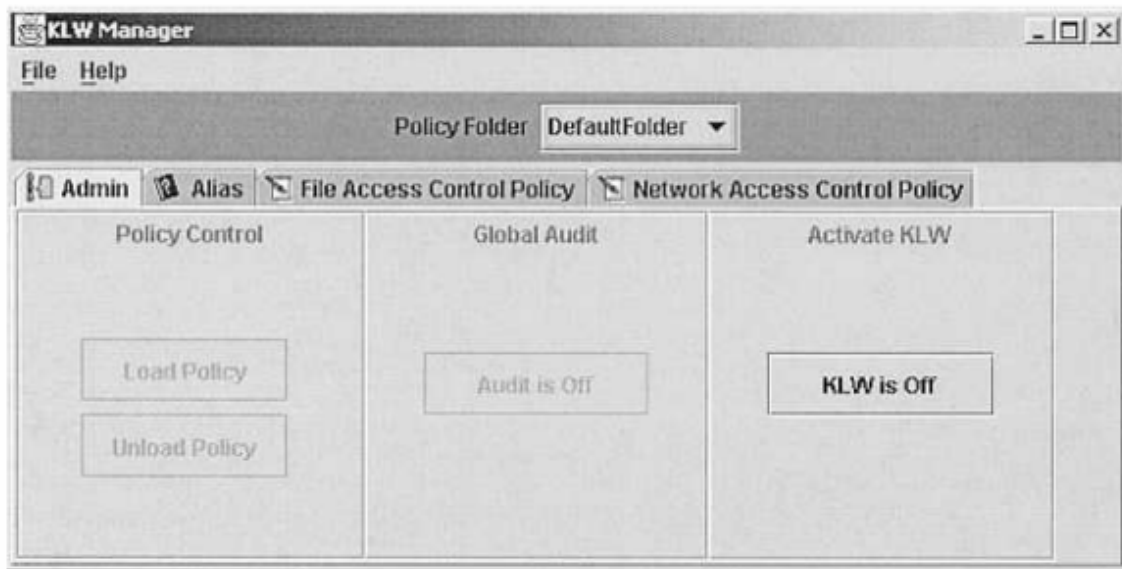
A.3.4.1 The "Load Policy" Button
The Load Policy Button will load whatever policy you have currently selected in the policy folder into the KLW Filter. The KLW Filter will start enforcing the loaded policy rules. This button is grayed out if you have not yet started KLW (See sections A.2.3 and A.3.4.4). If you change the contents of a policy folder, you need to press "Create Database" in the Alias tab to update the database. And pressing the Load Policy Button again will load the new policy. An audit message will be logged in the audit file when a new policy is loaded.

A.3.4.2 The "Unload Policy" Button
The Unload Policy Button will cause the KLW Filter to unload the current enforced policy. Unloading the policy causes the computer to behave as though the KLW Filter is not active. That would give permissions of file and network access to every request. This button is grayed out if you have not yet started KLW (See sections A.2.3 and A.3.4.4).

A.3.4.3 The "Audit is Off" Button

The Audit is Off button toggles the Global audit mode. By default KLW starts with global audit off. With global audit off no audit will be generated by the KLW filter, no matter what audit options may be set in the Alias, File or Network tabs (See sections A.3.5, A.3.6, A.3.7 and the Tutorial). In order for those audit options to be active you have to click this button and turn global audit on. Once you have clicked it, it will toggle to Audit is On. Clicking it a second time will toggle audit off again. This button is grayed out if you have not yet started KLW (See sections A.2.3 and A.3.4.4).

A3.4.4 The "KLW is Off" Button

Clicking the KLW is Off button will turn KLW on. This activates the KLW Filter by loading the KLW drivers to the system. But it does not load a policy. In order to load a policy you need to click the Load Policy Button (See section A.3.4.l). Once you have activated KLW Filter, it cannot be unloaded without restarting your computer. After you click KLW is Off, it will toggle to KLW is On and gray out. To deactivate policy enforcement without restarting your computer, you need to click the Unload Policy button (See section A.3.4.2).

A.3.5 Alias Tab
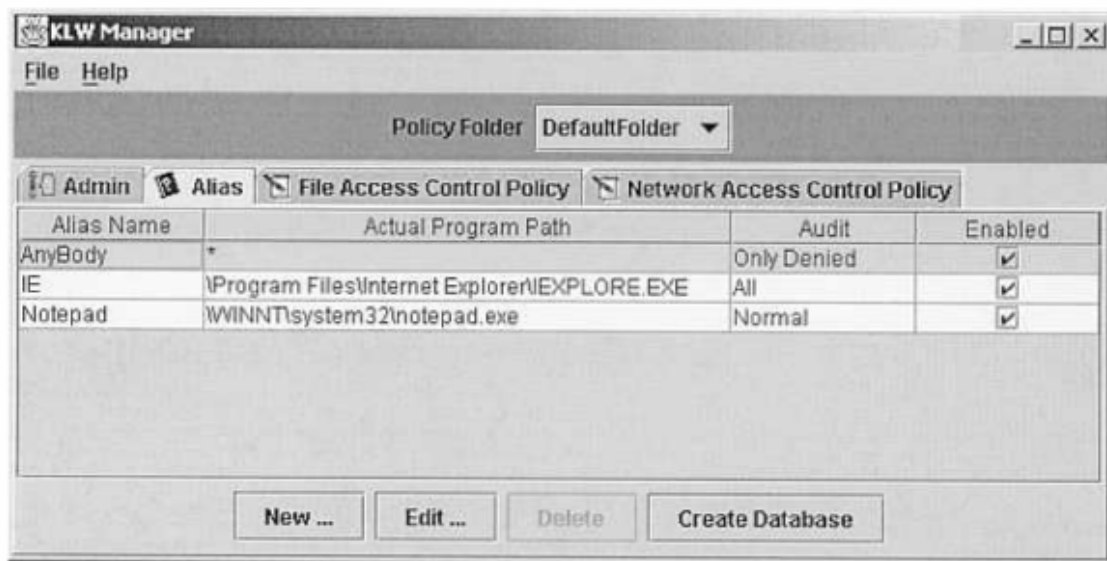
A.3.5.1 Current Policy Folder Combo Box

When you create a new policy you are actually creating a number of files which are all stored in that policy's folder. Thus when you want to switch to a different policy (say from Netscape to Media Player) you switch from the Netscape Policy Folder to the Media Player Policy folder. When you save a new policy the KLW Manager automatically puts the policy folder into the correct path and this combo box will always display all available policy folders. Just select the policy you wish to work with from the pull down menu and the KLW Master will display all that policy's information. You can only load

one policy folder at one time but each policy folder can hold policy for a number of executables.

A.3.5.2 Alias Name Field

An alias is just a shorter name for an executable used for your convenience. Thus C:\Program Files\Windows Media Player\mplayer2.exe can be aliased as MPlayer. This field displays the names of the various aliases you have defined.

A.3.5.3 Actual Program Path Field

This field displays the actual program path of the executable for which you have created an alias.

KLW supports the ability to alias a program depending on how the program was initiated. So, for example, you could alias MS Word so that when it is started from Netscape, it is treated differently than when it is started directly. This allows you to write policies that monitor a program such as Netscape as well as any programs that are started out of Netscape. The syntax for identifying a program started from another program is:

ExecutablePathOfProgramBeingAliased <- ExecutablePathOfThitiatingProgram.

The ns6Acro alias in the ns6 policy that comes with this distribution is an example of this.

A.3.5.4 Audit Field

This displays the level of audit you have selected. The various levels of audit are explained under Section A.3.5.6.1 Create a New Alias Dialog.

A.3.5.5 Enabled Field

Checking this field enables the policy associated with that alias, and deselecting it disables the policy. Thus, if you want to enforce only the Media Player policy, select Enabled for MPlayer and deselect it for all other aliases.

A.3.5.6 New... Button

Selecting the New Button opens the Create a New Alias dialog.

A.3.5.6.1 Create a New Alias Dialog

A.3.5.6.1.1 Alias Field

Enter the name of your alias here. For Media Player in the above example you would enter MPlayer.

A.3.5.6.1 .2 Absolute Path Field

You can type in the absolute path here or use the Browse Path... Button.

A.3.5.6.1.3 Audit Mode Combo Box

You can select one of the audit-mode options from the combo box below.

```
Normal
Only Granted
Only Denied
All
```

A.3.5.6.1.3.1 Normal

This sets the alias to be audited as the policy file says it should. Given a policy that says to audit all attempts of MPlayer to access C:\WINNT, which it is allowed to do, but not audit MPlayer's attempts to access C:\, which it is also allowed to do, and to audit all attempts to access C:\Program Files, which it is NOT allowed to do, attempts to access C:\WINNT and C:\Program Files will be audited and attempts to access C:\ will not.

A.3.5. 6. 1.3.2 Only Granted

This only displays audit when the aliased program attempts to do something it is allowed to do, and which the policy says to audit. Given a policy that says to audit all attempts of MPlayer to access C:\WINNT, which it is allowed to do, but not audit MPlayer's attempts to access C:\, which it is also allowed to do, and to audit all attempts to access C:\Program Files, which it is NOT allowed to do, only attempts to access C:\WINNT will be audited. If MPlayer tries to access something it is not allowed to access, it will do so silently.

A.3.5.6. 1.3.3 Only Denied

This only displays audit when the aliased program attempts to do something it is not allowed to do. Given a policy that says to audit all attempts of MPlayer to access C:\WININT, which it is allowed to do, but not audit MPlayer's attempts to access C:\, which it is also allowed to do, and to audit all attempts to access C:\Program Files, which it is NOT allowed to do, only attempts to access C :\Program Files will be audited. If MPlayer tries to access something it is allowed to access, it will do so silently.

A.3.5.6.1.3.4 All

This tells KLW to report everything an alias does no matter what the policy says. Given a policy that says to audit all attempts of MPlayer to access C:\WINNT, which it is allowed to do, but not audit MPlayer's attempts to access C:\, which it is also allowed to do, and

to audit all attempts to access C:\Program Files, which it is NOT allowed to do, ALL attempts would generate audit, (even C:\,) as long as they are all under the MiPlayer alias.

A.3.5.6.1.4 OK Button

Clicking this button will save the alias you have created.

A.3.5.6.1.5 Cancel Button

Clicking this button will exit the dialog without saving anything you have done.

A.3.5.6.1.6 Browse path... Button

Clicking this button will bring up an explorer-like window with which you can select the absolute path of the alias you are creating.

A.3.5.7 Edit... Button

This opens the Create a New Alias Dialog for an alias you have already created allowing you to edit any of the information you entered.

A.3.5.8 Delete Button

This button will delete an alias. If the alias you are deleting is used in any policy files, all references to it will be deleted when you delete the alias.

A.3.5.9 Save Button

This will save any changes you have made to the alias table. Before you press this button any addition or editing of aliases is not final.

A.3.5.1O Create Database Button

Pressing this button will do all the behind the scenes work to ready your final policy for loading into the KLW Filter. You must always press this button before pressing Load Policy to be sure the Manager has created a policy file to be loaded and that it has the latest changes you have made to the policy in it.

A.3.6 File Access Policy Tab

28
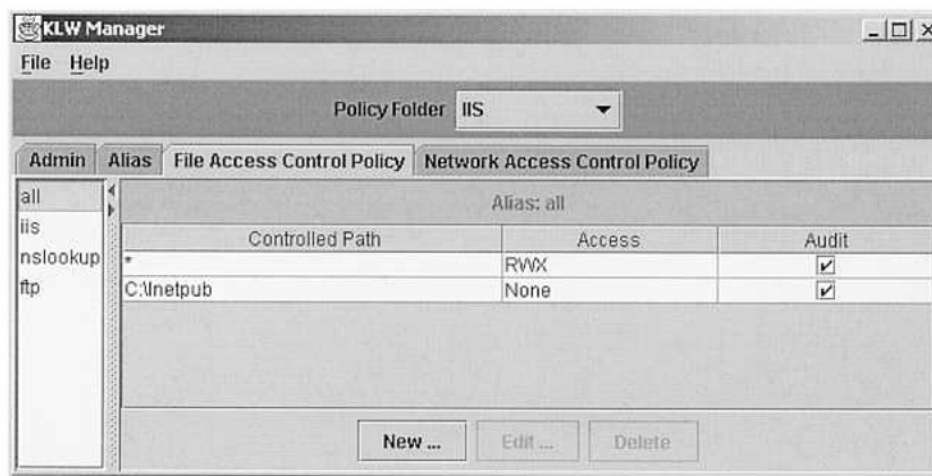
A.3.6.1 The Alias Pane

Within this pane you can select which alias's policy you want to edit. Each alias has its own set of policy which must be edited independently.

A.3.6.2 Controlled Path Field

This displays the path for which the alias is being controlled. In the above policy the "all" alias is controlling all program's access to C:\Inetpub.

A.3.6.3 Access Field

This displays which type of access the alias has to the controlled path. For C;\Inetpub all programs have No access. (The IIS program will be given access to this directory with a separate rule under the "iis" alias.)

A.3.6.4 Audit Field

Select this field if you want to audit the rule in question and deselect it if you do not want to audit the n in question. This is subject to how the alias's audit level is set (See section A.3 .5.4) and how the global audit is set (See section A.3.4.3).

A.3.6.5 The New... Button

Selecting the New... Button opens the Create a New Rule for File Access Control Dialog.

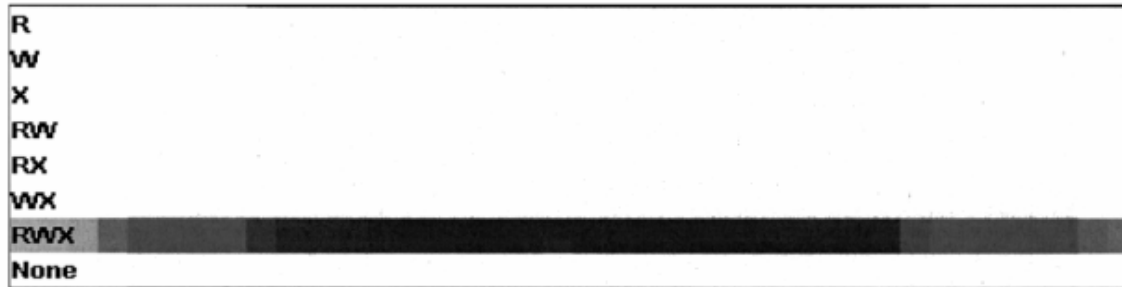### *A.3.6.5.l The "Create a New Rule for File Access Control" Dialog*

A.3.6.5.1.1 Controlled Path Field

Enter the path you wish to control here.

A.3.6.5.1.2 Access Level

The various access levels are R = Read, W = Write, X = executable. These can be combined as shown. You can also set access to None to disallow any access.

A.3.6.5.1.3 Audit Checkbox

Check this box if you want this rule to be audited.

A.3.6.5.1.4 OK Button

Click the OK button to save the rule you have created.

A.3.6.5.1.5 Cancel Button

Click the cancel button to exit the dialog without saving.

A.3.6.6 The Edit... Button

The Edit... Button opens the Create a New Rule for File Access Control Dialog with the currently selected rule's information in it. Changing the information adjusts the rule.
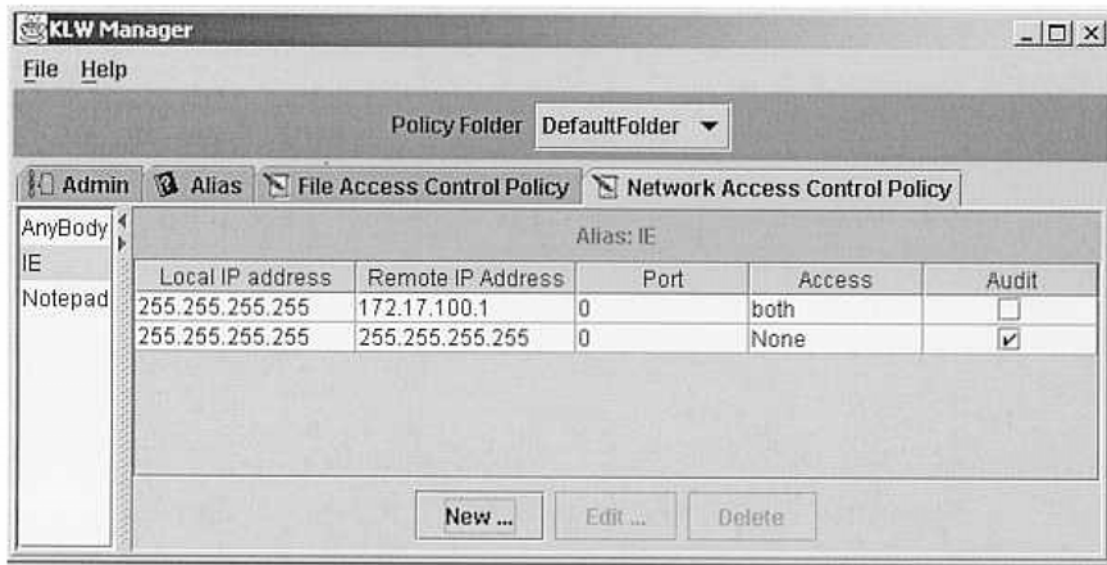
A.3.6.7 The Delete Button

This allows you to delete a rule from your policy.

A.3.6.8 The Apply Button

This saves the changes you have made to the alias's policy to its policy file. You still need to click the Create Database button in the Alias tab before you can Load the finished policy.

A.3.7 Network Policy Tab

A.3.7.1 The Alias Pane

Within this pane you can select which alias's policy you want to edit. Each alias has its own set of policy, which must be edited independently.

A.3.7.2 Local IP address Field

This field displays the local IP for the Network rule.

A.3.7.3 Remote IP Address Field

This field displays the remote IP for the Network rule.

A.3.7.4 Port Field

This field displays the destination port for the Network rule. For outgoing traffic this is the port on the remote machine. For incoming traffic this is the local port.

A.3.7.5 Access Field

This field describes whether the rule applies to incoming packets, outgoing packets, or both.

A.3.7.6 Audit Field

Select this field if you want to audit the rule in question and deselect it if you do not want to audit the rule in question. This is subject to how the alias's audit level is set (See section A.3.5.4).

A.3.7.7 The New... Button

Selecting the New... Button opens the Create a New Rule for Network Access Control Dialog.

A.3.7.7.1 The Create a New Rule for Network Access Control Dialog
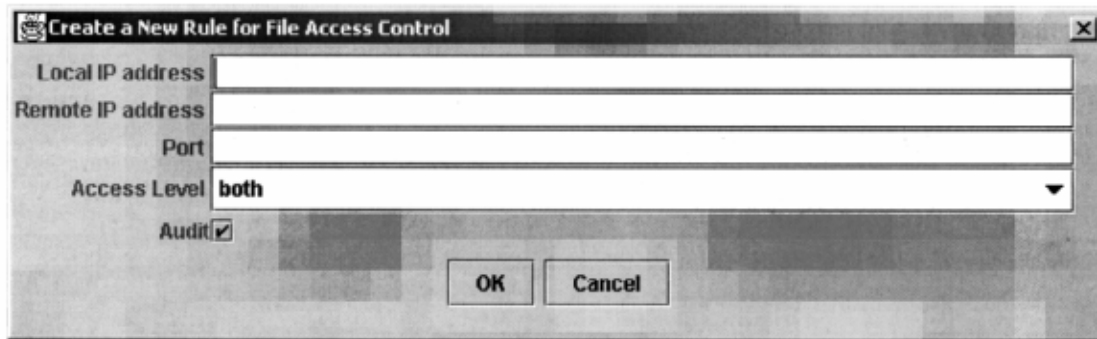
A.3.7.7.1.1 Local lPAddress Field

Enter the local IP to be controlled here.

A.3.7.7.1.2 Remote IP Address Field

Enter the remote IP to be controlled here.

A.3.7.7.1.3 Port Field

Enter the port to be controlled here.

A.3.7.7.1.4 Access Level
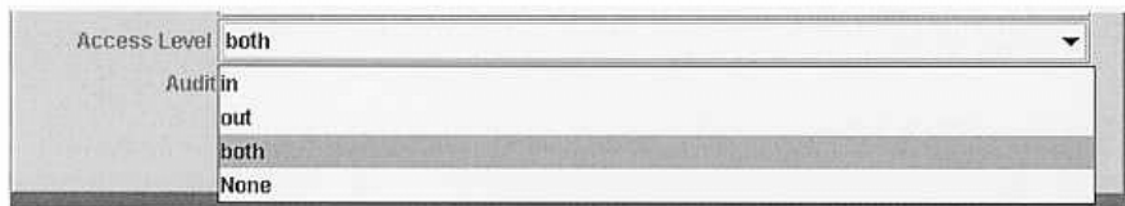
A.3.7.7.1.4.1 The Access Levels

The access level describes whether the rule applies to incoming packets (in), outgoing packets (out), or both. You can also set access to None to shut off certain ports or IPs in either direction.

A.3.7.7.1.5 Audit Checkbox

Check this box if you want this rule to be audited.

A.3.7.7.1.6 OK Button

Click the OK button to save the rule you have created.

A.3.7.7.1 .7 Cancel Button

Click the cancel button to exit the dialog without saving.

A.3.7.8 The Edit... Button

The Edit... Button opens the Create a New Rule for Network Access Control Dialog with the currently selected rule's information in it. Changing the information adjusts the rule.

A.3.7.9 The Delete Button

This allows you to delete a rule from your policy.

A.3.7.1O The Apply Button

This saves the changes you have made to the alias's policy to its policy file. You still need to click the Create Database button in the Alias tab before you can Load the finished policy.

A.4 Tutorial

A.4.1 Creating a File Policy for Media Player

A.4.1.1 Start KLW

From the Start Menu select KLW. Under KLW select KLW Manager. The KLW Manager will open to the Admin tab.

A.4.1.2 Turn the KLW Filter On

In the Admin tab of the KLW Manager click the "KLW is Off" button. This will start the KLW Filter.

A.4.1.3 Turn Audit On

In the Admin tab of the KLW Manager click the "Audit is Off" button. This will activate audit for the KLW Manager. You can turn audit off again by clicking the same button that will have toggled to "Audit is On".

A.4.1.4 Create the All Alias

Click the Alias tab of the KLW Manager. Click the New... Button at the bottom of the window. A new dialog will open. In the Alias field type All. This will be the alias which represents any given program or path. So if you wanted to write a rule that affects all programs you would use the All alias. In the Absolute Path field type a . This is the wildcard expansion character that tells the policy that any path matches. The last thing you need to do is select the audit level from the Audit Mode combo box. For the All alias you will want to select Normal audit. After you have selected the audit mode click OK. Once you are back at the Alias tab be sure to click the Save button. None of the changes you have made will take effect unless you do so.

A.4.1.5 Create the Media Player Alias

Click the Alias tab of the KLW Manager. Click the New... Button at the bottom of the window. A new dialog will open. In the Alias field type MPlayer. This will be the alias for Media Player throughout your policy. If you know it, you can type the path to the Media Player .exe file in the Absolute Path field. Otherwise you can click the Browse Path... button and use the explorer-like interface to find the executable. Once you have located the executable click the Open button and the path will appear in the Absolute Path field.

The last thing you need to do is select the audit level from the Audit Mode combo box. You will want to select Only Denied as the audit mode when you are first generating a policy for a program. The reason for this is that you are going to deny the program any access, and then see what it tries to do via the audited failures. For example — one of the first things Media Player tries to do is get read access to the file you are trying to view. Since we have denied Media Player any access to anything the audit will show a failed attempt to get read access to said file. You can then give it read access to that file and then run it again, seeing what new failures arise. In this fashion you generate a least-privileged policy.

The only wrinkle in this plan is that sometimes a program will ask for more than it needs. For example Media Player will ask for read access to all of the C:\ drive first, when it really only needs to read the file it is trying to play. So be aware of this fact when creating your policy. After you have selected the audit mode, click OK and your alias will be created. Again — remember to click the Save button on the Alias tab.

A.4.1.6 Add the First Three File Rules

Every file policy should begin with the same first three rules.

1. Everything is allowed all privileges to everything.

2. The alias in question (here MPlayer) is not allowed any privileges to anything.

3. Nothing is allowed any privileges to the alias in question (here MPlayer).

Without the first rule nothing on your system would work. As soon as you started KLW the entire computer would be so thoroughly locked down you wouldn't even be able to restart. The first rule creates a "wide-open" policy. The second rule locks down the alias in question. In order to create a least-privileged policy you have to deny the alias any permissions and then add rules to give it only the access you want it to have. The third rule keeps someone from overwriting your executable with malicious code.

To generate these rules for your MPlayer policy go to the File Policy tab of the KLW Manager. Start with rule 1. In the alias list on the right-hand side select the All alias. Then click the New... button at the bottom of the window. The new file rule dialog will open. In the Controlled Path field type a 'K In the Access Level combo box select RWX. Deselect the audit checkbox, you do not want the entire system's file accesses muddling up your audit window. Click OK and you have created the first rule.

All; * ; RWX — everything is allowed all privileges to everything.

Next, with the All alias still selected, click on New... again. This time in the Controlled Path type the path to MPlayer's executable if you know it or click Browse Path... and find it via the explorer interface. For access level select None and deselect the audit checkbox. Click OK and you have created rule 2.

> All ; C:\Program Files\MPlayer.exe ; None — Nothing is allowed any privileges to the alias in question.

Finally in the alias list on the right select the MPlayer alias and click New. In the controlled path type * and in the Access Level select None. Make sure the audit checkbox IS selected, click OK and you've created rule 3.

> MPlayer; * ; None — The alias in question is not allowed any privileges to anything. You want to be sure to audit this rule so that you will be able to generate your MPlayer policy from the access it denies.

After you have created these three rules click Apply. Be sure to always click Apply after generating any rules as the policy will not be updated if you don't.

A.4.1.7 Generate Audit from the First Three Rules

Now that you have the first three rules in place you are ready to generate some audit. First go to the Alias tab and click Create Database. You must always click Create Database before you attempt to load a policy as the internal files needed to load your policy are not created until you do. Once you have clicked Create Database start DebugView from the KLW root directory. This will activate DebugView which acts as the KLW audit viewer. Next click the Load Policy button in the Admin tab, this begins enforcing the policy you have created immediately (since you have already started the KLW Filter). Once you have done this simply try to open a file with Media Player. You will notice some audit scrolling across DebugView and Media Player will fail (it does not yet have the needed permissions to start). Save the audit from your audit viewer and search for any instances of the word "denied". Any attempt Media Player made to access a file will be denied (due to rule #2). In order to start Media player you will want to give it access to everything it tries to use when starting.

A.4.1.8 Write your File Policy

Now that you have your first bit of audit you are ready to start writing your policy. Look through all the denied messages relating to MPlayer. For each denied path create a new rule for the MPlayer alias. For example — if MPlayer tried to get RX access to C:\WINNT\ocx.dll create a rule allowing MPlayer read and execute access to that path. Sometimes an executable will ask for both RX to C:\WINNT and RX to C:\WINNT\ocx.dll. Give it only the access it needs to function. In the case of Media Player it won't function unless you give it read access to all of the C:\ drive. Unless you do not mind it reading all your systems files what you need to do is give it read to all of C:\ and then create rules blocking access to the portions of C:\ you don't want it to be able to read from, like C:\Top Secret\Moon Laser Plans. After you have created the rules based upon the audit Save them, Create Database again, Load Policy again, and try starting Media Player. You will get a new set of audit that will lead to a new set of rules. Keep following this procedure until the program is as locked down as you need it or there

are no more denied messages. Another method of generating audit is to change rule two to allow Media Player to do anything it likes. Then search your audit logs for any instance of Media Player, you will find a great number of allowed access checks. Turn all of these allowed access checks into rules and then re-apply rule 2, disallowing Media Player any access say what it needs to start. Either way you will end up with a secure policy for Media Player which you can then adjust to your personal needs.

A.4.2 Creating a Network Policy for Media Player

A.4.2.1 Add the First Three Network Rules

Use the same methods you did for the file policy. For rule 1 the All alias should have a * in Local and Remote IP as well as port number with an access level of both and audit off. Rule 2 will be for the MPlayer alias and will have a * in both IP fields and the port field, the access will be none, however. Rule 3 is already there — it is the same for both file and network policies.

A.4.2.2 Generate Audit from the First Three Rules

Now that you have the first three rules in place you are ready to generate some audit. Just as with file permissions first try to access the remote MPlayer file with no permissions. This will generate denied messages in your audit.

A.4.2.3 Write your Network Policy

This is exactly the same as the file policy for Media Player, save that you are checking IP and port addresses as opposed to file paths. Another thing to remember with network rules is that unlike file rules, which search for the longest possible path-name/rule match, the network rules simply use the first match they find. So if you leave your 'anything can do everything' rule at the top of your policy the more restrictive rules will never be hit — every access check will match the first rule and stop there. So for network policies you need to order your rules from most restrictive to least restrictive. Keeping this in mind generate your least privileged policy using your audit just as you did for file access.